AFRL-RI-RS-TM-2008-17
Final Technical Memorandum
April 2008

HIGH PERFORMANCE EMBEDDED COMPUTING SOFTWARE INITIATIVE (HPEC-SI) PROGRAM FACILITATION OF VSIPL++ STANDARDIZATION

Georgia Tech Applied Research Corporation

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TM-2008-17 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
STANLEY LIS Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
APR 2008	Final	Jun 06 – Dec 07

4. TITLE AND SUBTITLE

HIGH PERFORMANCE EMBEDDED COMPUTING SOFTWARE INITIATIVE (HPEC-SI) PROGRAM – FACILITATION OF VSIPL++ STANDARDIZATION

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-06-1-0179

5c. PROGRAM ELEMENT NUMBER
63755F

6. AUTHOR(S)

Daniel P. Campbell

5d. PROJECT NUMBER
HPEC

5e. TASK NUMBER
01

5f. WORK UNIT NUMBER
08

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Georgia Tech Applied Research Corporation
Centennial Research Building
Georgia Institute of Technology
Atlanta, Georgia 30332

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RITB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TM-2008-17

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 08-2515

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Vector Signal and Image Processing Library (VSIPL) is an industry standard Application Programming Interface for embedded signal processing tasks. The High Performance Embedded Computing Software Initiative (HPEC-SI) program is a collaborative program to establish extensions to the VSIPL specification to support Object Oriented elements of the C++ programming language, and encapsulated support for data parallel processing. The program goals include the simultaneous threefold improvement in software portability, threefold improvement in developer productivity, and fifty percent improvement in software performance compared to standard practices. This report describes the efforts of the Georgia Tech Research Institute in support of the HPEC-SI program during the period from June 2006 through December 2007. These efforts included development of organizational strategies, participation in the HPEC-SI Applied Research and Development Working Groups, dissemination of program results to outside organizations via internet tools, and maintenance of a parallel computing software testbed for program participants, and development of an automated process for the configuration of certain types of computing nodes.

15. SUBJECT TERMS
VSIPL C++ Object Oriented High Performance Software

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Stanley Lis
U	U	U	UL	13	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

Figure **Page No.**

1. Introduction

During the period of 1 June 2006 through 01 December 2007, the Georgia Tech Research Institute (GTRI) engaged in the project "Facilitation of VSIPL++ Standardization," in support of the High Performance Embedded Computing Software Initiative (HPEC-SI) Program. GTRI contributed to the HPEC-SI forum objectives by assisting in the development of Object Oriented VSIPL standards, co-chairing the HPEC-SI Development Working Group, assisting in the dissemination of technical designs and ideas, and the maintenance of a parallel computing testbed for VSIPL++ experiments by HPEC-SI program participants.

2. Tasks Completed

GTRI provided specific standards prototypes for the Working Group for discussion and implementation. GTRI served on the Technical Advisory Board of the HPEC-SI program. GTRI served as co-chair for the HPEC-SI Development Working Group. GTRI maintained the website[2] for the HPEC-SI effort, which served as a collection point for information about the effort, as well as information presented at the meetings, for the forum members. GTRI maintained and made available for program participants a parallel computing testbed.

GTRI attended HPEC-SI working group meetings in August 2006, January 2007, and June 2007 in support of program efforts. At these meetings, GTRI participated in discussions vetting the proposed VSIPL++, parallel VSIPL++, and proposals to expand the parallel VSIPL++ specification to support heterogeneous and tiled computing architectures.

GTRI continued development and maintenance of the HPEC-SI Program website, including meeting notes and presentations for each of the Working Group meetings during the project period, conference presentations, and prototype applications. In conjunction with this, GTRI continued to maintain email reflectors for use by HPEC-SI program participants.

GTRI made a parallel computing cluster available for use by HPEC-SI program[3] participants as a testbed for VSIPL++, parallel VSIPL++, and other parallel computing systems. The cluster is a fifty five node Beowulf style cluster with 116 compute processors of varying types. Four of the compute nodes are based on the IBM/Toshiba/Sony Cell Broadband Engine, which is a platform of focus for the HPEC-SI program. Several HPEC-SI program participants were given or maintained login access during the project and used the testbed. Software support, including the installation of software and tools useful to the HPEC-SI program was continued.

GTRI continued to maintain the VSIPL++ User's Guide[4]. The Guide is intended to complement the VSIPL++ Specification Document and serve as an introduction and clarification of the Specification for application programmers. The guide contains elaborations and examples from portions of the VSIPL++ Specification that the HPEC-SI Working Groups find to be difficult or confusing for new VSIPL++ application programmers. The choices of topics draw heavily from the experiences of the projects undertaken by the Demonstration Working Group, as well as the vetting and clarification of the VSIPL++ Specification by the Development Working Group.

GTRI continued its advisory role on the Technical Advisory Board of the HPEC-SI program, as well as serving as the co-chair of the Development Working Group.

3. Results

3.1 Heterogeneous & Tiled Parallel extensions to VSIPL++

GTRI participated in discussions leading up to the conceptual design of extensions to the Parallel VSIPL++ specification[5] designed to support heterogeneous computing accelerators, as well as tiled chip multiprocessors. Examples of targeted tiled architectures include those developed under the DARPA Polymorphous Computing Architectures program, the IBM/Sony/Toshiba Cell Broadband Engine, and commodity multicore general purpose processors. GTRI was not the primary author of these extensions, but nevertheless was actively involved in the preliminary discussions in anticipation of including such functionality in future versions of the VSIPL++ specification.

Extensions to Parallel VSIPL++ primarily focused on the addition of task and communication abstractions. These abstractions decouple the execution of portions of the application (tasks) and define an interface mechanism (conduits) between the tasks. This decoupling allows execution independence among the tasks, which in turn allows late binding of tasks to actual compute hardware, more efficient utilization of each hardware element, and a productive, abstract method of defining data dependencies between tasks at the application level.

3.2 Cluster-friendly Playstation3 Unattended Software Build Process

GTRI inserted several STI Cell Broadband Engine based compute nodes, provided by the government, into the parallel software testbed for use by HPEC-SI program participants. These compute nodes took the form of Sony Playstation3 (PS3) game consoles. Each console contains a single Cell processor, with 7 functioning Synergistic Processing Elements (SPEs). The Sony-provided firmware for the PS3 which allows users to install third party operating systems that have access to 6 of the 7 SPEs, as well as the general purpose portion of the Cell processor. The Linux kernel has been ported to support the Cell processor, and many distributions of the Linux operating system have been ported to easily install on, and fully exploit the Cell processor as available from a third party operating system. In order to support the goals of the HPEC-SI program, the government supplied four PS3 consoles, which GTRI inserted into to the parallel software testbed.

A computer that is used as a compute node in a Beowulf-style cluster requires a means of automated, unattended, definable wipe and reinstall of system software. This is required in order to guarantee consistency between compute nodes, to facilitate system-wide configuration changes, and to allow efficient maintenance and troubleshooting of the nodes. This capability is widely available for computers that are frequently used in such a manner, but is not for the PS3. All of the widely available operating system installation packages require human intervention and selection of configuration options.

In order to allow cost effective administration of the PS3 compute nodes, as well as facilitate future upgrades and software installation, GTRI developed an unattended, automatic, configurable system wipe and reinstall process for the PS3. A cursory description of this process is currently available electronically (Reference 6).

The specific process that GTRI developed is customized to the Parallel Software Test and Evaluation Center (PaSTEC) testbed, but is extendable to other environments. An overview of the technique used is described in the following paragraphs.

A modified operating system boot loader (otheros.bld) for the PS3 is built and installed. This boot loader will query a specified Trivial File Transfer Protocol (TFTP) server for additional Linux kernel options that are specified by the system administrator for the network address of the PS3 being booted. If these instructions are not present, the PS3 will boot normally. If present, the additional kernel options can be set to instruct the PS3 to load and execute the Kickstart process. Kickstart is an automated version of the Anaconda installer for Red Hat derived Linux distributions. In order to maximize compatibility with Cell processor tools and Cell VSIPL++ development, GTRI elected to install the Fedora Core 7 (a Red Hat derived distribution) operating system on the PS3s. The Kickstart program then loads further configuration details over the cluster local network from the main server, formats the node-local hard drive, and builds the distribution on the PS3. This process allows system administrators to set a common configuration file for all nodes of a given type, and to directly initiate the rebuild of any node, from software controls.

Several elements of the Kickstart program were incomplete or faulty for the Fedora Core 7 distribution when used in this manner. GTRI found and enabled a workaround for each of these problems. A more detailed description of the steps required is available via the World Wide Web[7]. This description was written as a high level, cursory guide for system administrators, a capture of this guide as of February 20, 2008 is shown in Figures 1 through 4. This process has allowed GTRI to deploy the PS3 nodes into the parallel software testbed rapidly, and maintain consistent configuration across all of the PS3 nodes, through several software upgrades. Using this process, the initial deployment of the four PS3 nodes into the PaSTEC cluster required approximately 30 minutes of operator time and 90 minutes of unattended time to complete.

PS3 Auto Install Process

From Cluster Administration Wiki

Contents

Concept of Operations

- This process is modelled loosely on the PXE/Netboot and Anaconda Kickstart automated install process widely used for "Normal" Fedora Core servers. The PS3 does not have quite the same netboot capabilities as a traditional cluster server, so we must engage in some tricks to mimic the behavior.
- Compile a customized kboot, which provides a new otheros.bld to be loaded onto the PS3
- Configure a DHCP server to provide specific addresses to known PS3 nodes
- Early in the boot process, after obtaining an IP address via DHCP, new kboot asks a site-specific tftp server if an alternate kboot.conf has been assigned to that IP address
  - If not, boot will proceed normally, skip the rest
- Special kboot.conf causes an install kernel and initrd.img to be downloaded via TFTP and sets the default launch option to a configuration that uses the install images, and points to a kickstart configuration file
- kickstart configuration file is obtained via NFS, contains anaconda build script, and points to FC7 media via HTTP
  - See ours (https://pastec.gtri.gatech.edu/ks-fc7-ps3.cfg) for a sample.
- FC7 media must be slightly modified to work with YDL installer. FC7 installer hangs at reboot.
- Kickstart script is fairly typical, with some exceptions:
  - Mounts /install from front node via NFS which contains various installation files
  - Installs all rpms found in /install/ps3-fc7-rpms directory (including an upgraded kernel - default FC7 kernel hangs on reboot attempt)
  - Copies SDK 3.0 installation files to local drive, and creates a link to installtion script in /etc/rc.d
  - Copies a bunch of cluster-wide common files, including /etc/passwd, /etc/fstab, /etc/rhosts, and the like.

Prerequisites

- PS3 already running linux, do one manual install
- DHCP server, configured to deliver static IP addresses to the PS3 nodes
- Server running http, nfs, and rsh/rcp servers that the node to be installed has access to.

Build customized kboot

- This step will build a custom kboot (otheros.bld) that will check a tftp server for alternate boot instructions

Figure 1. PS3 Auto Build Guide Page 1

- download modified kboot source here (http://ftp.riken.go.jp/pub/Linux/kernel/people/geoff/cell/ps3-linux-distro-kit-20061208/src/kboot-20061208.tar.gz)
- Unpack onto ps3 already running YDL or FC

- Ensure that you have all the pre-requisites required to compile kboot.
- from the kboot-10 directory run "make download"
- from the top level kboot directory, run "make".
- when the make is complete, create a special kboot-10/root/etc/kboot.conf to ask tftp server for a node-specific kboot.conf on boot, if available. Here is the one I use:

```
root=LABEL=/

ipaddr=`/sbin/ifconfig eth0 | grep "inet " | awk '{split($2,add,":");split(add[2],addr,".");printf("%02X%02X%02X%
02X\n",addr[1], addr[2], addr[3], addr[4])}'`
tftp -l /foo -r pxelinux.cfg/$ipaddr -g 130.207.222.99 2> /dev/null

default=ydltext
timeout=5
root=/dev/sda1
ydl='/dev/sda1:/vmlinux-2.6.16-20061110.ydl.2ps3 initrd=/dev/sda1:/initrd-2.6.16-20061110.ydl.2ps3.img root=/dev/sda2
init=/sbin/init video=ps3fb:mode:3 rhgb'
ydl480i='/dev/sda1:/vmlinux-2.6.16-20061110.ydl.2ps3 initrd=/dev/sda1:/initrd-2.6.16-20061110.ydl.2ps3.img root=/dev/sda2
init=/sbin/init video=ps3fb:mode:1 rhgb'
ydl1080i='/dev/sda1:/vmlinux-2.6.16-20061110.ydl.2ps3 initrd=/dev/sda1:/initrd-2.6.16-20061110.ydl.2ps3.img root=/dev/sda2
init=/sbin/init video=ps3fb:mode:4 rhgb'
ydltext='/dev/sda1:/vmlinux-2.6.16-20061110.ydl.2ps3 initrd=/dev/sda1:/initrd-2.6.16-20061110.ydl.2ps3.img root=/dev/sda2
init=/sbin/init 3'

eval `cat /foo`
```

- - Note on above: I believe the ydl lines are unnecessary but have not tested. My installation process installs a new /etc/kboot.conf on the node after build, and this is used if the tftp fails.
  - modify the tftp line as necessary to reach your local server.
- run make again
- from the kboot top level directory, execute:

```
gzip -9 < kboot-10/linux > otheros.bld
```

- Copy otheros.bld to USB drive and install as per the usual PS3 installation process.

Configure DHCP Server

- Obtain the MAC address of each PS3 to be installed
- Configure the DHCP server to delver a static IP address to the PS3

Configure tftp server

- Alternative kboot.conf will be requested from the compute node
- The boot loader will look for a file that has the name of the hex representation of the node's IP address
  - e.g. "0AFFFFC7" is the file for 10.255.255.199
  - We have a set of kboot files for the different node types on our cluster and a script (https://pastec.gtri.gatech.edu/rebuild) that creates a soft link to the appropriate filename
  - The last step in the kickstart process removes this file from the server, so that subsequent reboots do not trigger an install
- If present, this file takes the place of the node's default kboot.conf
- Below is the file used on this system

Figure 2. PS3 Auto Build Guide Page 2

```
default=installation
installation='//kernel ks=nfs:130.207.222.99:/install/kickstart/ks-fc7-ps3.cfg initrd=//initrd.gz video=480i init=/sbin/
init 3'
out1=`tftp -g -l /initrd.gz -r initrd-ydl-ps3.img.gz 130.207.222.99 > /dev/null`
out1=`tftp -g -l /kernel -r vmlinuz-ydl-ps3 130.207.222.99 > /dev/null`
unset out1
unset foo
```

- - note the method for obtaining the kickstart file (nfs). We found that for the YDL5 installer, obtaining the kickstart configuration file via http led to a crash
- place the specified boot files in the tftp directory
  - In the above example, we use the initrd and vmlinuz that come from the Yellow Dog Linux 5 installation boot media

Provide kickstart config file

- See modifications below
- place a kickstart configuration file in the location specified by the "ks=" clause in the kboot.conf file
  - For us this is on NFS share /install in a kickstart directory with a specific kickstart config file for each node type
  - You can see our directory structure [here (https://pastec.gtri.gatech.edu/install/)] - it is served via NFS as well as HTTP/HTTPS
- A good way to obtain a baseline kickstart config file is to do a manual install, and then grab the file "/root/anaconda-ks.cfg"
  - Several modifications are required to integrate into the cluster environment, and to install the Cell SDK, see below

Modify and serve FC7 media

- The YDL5 installer can not install from the raw FC7 media, but slight modifications allow it work
- Download and unpack a FC7 install image into a directory
- from FC7 root directory, "mkdir YellowDog"
- "cd YellowDog"
- "ln -s ../Fedora RPMS"
- "mkdir base; cd base"
- copy "minstg2.img" from a YDL5 install disk "YellowDog/base/minstg2.img"
- "cp ../../images/stage2.img ."
- link or copy the FC7 directory to the web server location specified in the kickstart config file

Customize kickstart config file

- Here (https://pastec.gtri.gatech.edu/install/kickstart/ks-fc7-ps3.cfg) is ours
- Customize package list to fit your needs
- Remember to set the encrypted root password as desired. We have one cluster-wide root password
- Notes on kickstart configuration:
  - must "mkdir /spu" so that the spufs can be mounted
  - we install kernel 2.6.23-rc3, because the FC7 stock kernel hangs on reboot
  - our cluster has a NFS Share on the front node at /install. Several steps in our kickstart config reference files there
  - We were not able to make installation of the Cell SDK work _during_ the OS install, so we use a hack to cause it to be installed the first subsequent time the system is booted:

Figure 3. PS3 Auto Build Guide Page 3

```
cp /mnt/temp/files/ps3-fc7-cellsdk-install /etc/rc.d/rc.sysinit.author
echo "if [ -f /etc/rc.d/rc.sysinit.author ] ; then /etc/rc.d/rc.sysinit.author; fi" >> /etc/rc.d/rc.local
chmod 755 /etc/rc.d/rc.sysinit.author
```

- Here (https://pastec.gtri.gatech.edu/ps3-fc7-cellsdk-install) is the ps3-fc7-cellsdk-install script. This script runs the cellsdk process. The http_proxy variables are set to allow the node to access the outside world during the yum steps. Without this, the process takes hours to complete for us. We run a http proxy on port 3148 of the front node.
- Install any rpms in the nfs:/install/rpms/ps3-fc7-rpm directory. We put any extra rpms that need to be installed for this platform in that location.
- Copy over the system kboot.conf file from the template on the file server
- Copy over the local yum repository to the node
- Call the ks-include script
  - This script contains the common, system-wide file copying done by all nodes of all types
  - nfs mount points, shadow and password files, group, hosts, fstab, ssh and rsh config files
  - remove the ip-specific install trigger file from the front node's tftp directory
  - Here (https://pastec.gtri.gatech.edu/install/kickstart/ks-include) is ours.
- install gmond
- turn on and off a few services

Figure 4. PS3 Auto Build Guide Page 4

4. Conclusions

During the period of performance of the subject contract, the Georgia Tech Research Institute contributed to the High Performance Embedded Computing Software Initiative program as a technical participant, and as a member of the technical advisory board. GTRI developed software strategies, assisted in the dissemination of program results via internet tools, provided a parallel software testbed for program participants, and developed a novel method for the unattended deployment of Linux-based operating systems onto PS3 compute nodes in a Beowulf-style cluster. GTRI also participated in technical advisory planning for the HPEC-SI program.

5. References

1. Schwartz, D. A ., Judd, R. R., Harrod, W. J., and Manley, D. P., "VSIPL 1.2 API" available electronically at http://www.vsipl.org/VSIPL_1p2_1.pdf.

2. HPEC-SI Web Site, http://www.hpec-si.org.

3. The Georgia Tech Research Institute Parallel Software Test and Evaluation Center – https://pastec.gtri.gatech.edu.

4. Bergmann, J., Emeny, S., Judd, R., Pancoast, R., Leimbach, D., Sacco, S., and Sroka, B., "VSIPL++ User's Guide," available electronically at http://www.hpec-si.org/VSIPL++%20User_s%20Guide%20Draft%20v0.2.pdf.

5. CodeSourcery, LLC, "VSIPL++ Specification – Parallel Specification," available electronically at http://www.hpec-si.org/spec-par-1.0-final.pdf.

6. PS3 Auto Build Guide, "Concept of Operations" available electronically at https://pastec.gtri.gatech.edu /admin-wiki/index.php/PS3_Auto_Install_ Process#Concept_of_Operations

7. PS3 Auto Build Guide, available electronically at https://pastec.gtri.gatech.edu/admin-wiki/index.php/PS3_Auto_Install_Process

List of Acronyms

API	Application Programming Interface
DARPA	Defense Advanced Research Projects Agency
GTRI	Georgia Tech Research Institute
HPEC-SI	High Performance Embedded Computing Software Initiative
PCA	Polymorphous Computing Architectures
PS3	Playstation3
SPE	Synergistic Processing Element
TFTP	Trivial File Transfer Protocol
VSIPL	Vector Signal & Image Processing Library